作者：知乎用户  
链接：https://www.zhihu.com/question/30315894/answer/154979413  
来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

**0. 前言**

本文面向初学者（但不是纯小白），分享一点我的经验。<del>毕竟百度“VS Code C”出来的第一条就是这个网页</del>现在不是了。其实VS Code真的不太适合写C，姑且算一种折腾吧。

本文所有内容均可从VS Code的官方文档：[C++ programming with Visual Studio Code](https://link.zhihu.com/?target=https%3A//code.visualstudio.com/docs/languages/cpp" \t "_blank) 以及各个扩展的文档中获得，并且他们还会进行更新。（本文也进行过二点五次重大更新）如果你还想了解得更深一点，可以去看。其实本文基本上是由不断地尝试得出来的，如果有错误可以指出。

我的环境：64位Windows 10。如果你是32位的，在某些地方需要进行修改，不过本文中没有提，可以自己试着改。现在的配置在Linux下应该也是可用的。

**1. 环境的准备**

VS Code的官网和下载、安装，我就不多说了。点那个箭头可以下到其他操作系统的版本，现在x64系统下的直接是64位的，不用点箭头。另外VSC每个月都会更新一次（自动下载更新），不要告诉我按照我的配置运行不了最后发现你用的是好几个月以前的版本。

VS Code只是一个编辑器，并不是IDE（集成开发环境）。不含编译器（和许多其它功能），所以编译器要自己安装好。如果想用其他工具链或单纯用官方扩展，参见第九点。

其实MinGW和MinGW-w64只是名字像，它们是两个不同的项目。为了方便，本文中的MinGW指的其实都是MinGW-w64。MinGW本身已经很久没有更新了，故**不推荐**。下载以下两个程序（都要）：

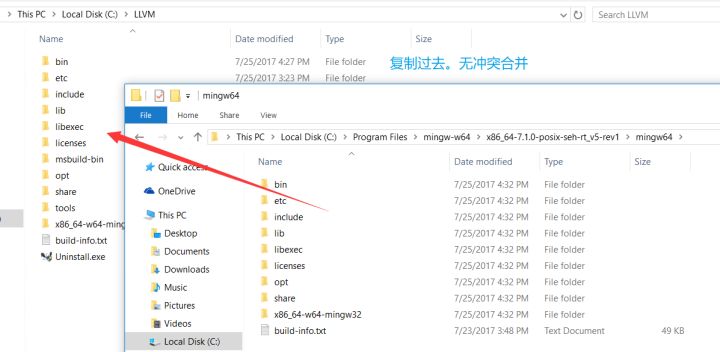
* [LLVM Download Page](https://link.zhihu.com/?target=http%3A//releases.llvm.org/download.html) 在此页面下载Clang。选Pre-Built Binaries中的Clang for Windows (64-bit)，**不需要下.sig文件**
* [MinGW-w64 - for 32 and 64 bit Windows](https://link.zhihu.com/?target=https%3A//sourceforge.net/projects/mingw-w64/) 在此页面下载MinGW-w64，点那个Download就行

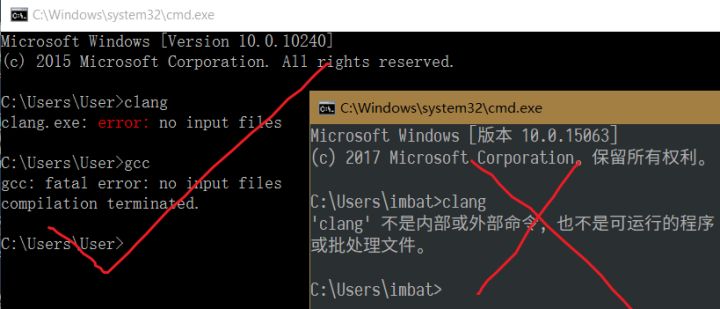
下载好了以后安装。添加环境变量时：选*Add LLVM to the system PATH for all users*（即第二项，不过第三项也差不多）。Clang的安装路径（Destination folder）我推荐填*C:\LLVM*，不装那里也行，下面的配置里路径就自己改。安装完了以后可能会弹出cmd说*MSVC integration install failed。*这个是因为Clang默认使用的是msvc的工具链，而我们选择的工具链是MinGW，所以就**不用管这个提示**。如果你想用别的工具链，参考第九点。

MinGW随便装哪，Architecture选x86\_64，装好以后把东西全部复制到Clang的文件夹里去，他们会**无冲突合并**，效果图见下。同样，不做这一步也行，下面的配置里路径就自己改，还要手动把MinGW的bin文件夹加到path中，因为MinGW不会自己加。至于为什么既要装Clang又要装MinGW，是因为Clang没有头文件。然后就可以把MinGW删了（Uninstall.exe）。**不建议安装多个MinGW**，如果你安装了其他IDE需要注意把其他的MinGW从环境变量中去掉；也可以自己把他们的编译器设为Clang。因为几乎所有的轻量级IDE用的都是MinGW或TDM-GCC，它们不制造编译器，只是打包了一个。而且它们用在VSC里也会有奇怪的错误。

如果因为网络问题一直下载失败，建议优先自己**想办法**（懂我的意思吧？）。实在不行，我提供一个我下好的MinGW（7.2.0版）：[https://pan.baidu.com/s/1slwTebV](https://link.zhihu.com/?target=https%3A//pan.baidu.com/s/1slwTebV" \t "_blank)

运行cmd，输clang，应该会提示*no input files*而不是“不是内部命令或外部命令”或者“无法将“clang”项识别为 cmdlet、函数、脚本文件或可运行程序的名称”。输clang -v或gcc -v可以显示出各自的版本。如果是“不是内部命令或外部命令”，说明clang.exe在的文件夹（我的是*C:\LLVM\bin*）没有在环境变量中，要加到path里才行。怎么做自己百度。如果加了还是这样，重启。

<img src="https://pic2.zhimg.com/v2-a3d0a7feee90411b98d8d227c83a6f25\_b.jpg" data-caption="" data-size="normal" data-rawwidth="1197" data-rawheight="585" class="origin\_image zh-lightbox-thumb" width="1197" data-original="https://pic2.zhimg.com/v2-a3d0a7feee90411b98d8d227c83a6f25\_r.jpg">

<img src="https://pic4.zhimg.com/v2-09f3ba476bdede18e9a9ab36ae8d71f3\_b.jpg" data-caption="" data-size="normal" data-rawwidth="819" data-rawheight="351" class="origin\_image zh-lightbox-thumb" width="819" data-original="https://pic4.zhimg.com/v2-09f3ba476bdede18e9a9ab36ae8d71f3\_r.jpg">

需要安装的扩展：

* C/C++（就是有些教程里的cpptools）
* C/C++ Clang Command Adapter：提供静态检测（Lint），很重要
* Code Runner：右键即可编译运行单文件，很方便

其他可选扩展：

* Bracket Pair Colorizer：彩虹花括号
* Include Autocomplete：提供头文件名字的补全
* C/C++ Snippets：Snippets即重用代码块，效果自己百度；这个扩展安装量虽高，不过个人感觉用处实在不大，你也可以选择其他的Snippets扩展甚至自己定义
* One Dark Pro：大概是VS Code安装量最高的主题
* vscode-clangd：这个和Adapter二选一，出得比Adapter晚，下载量也低，但却是llvm官方出的。出现问题时可以换着试试

不建议/不需要装的扩展：

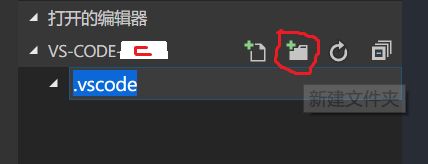
* GBKtoUTF8：把GBK编码的文档转换成UTF8编码的。此扩展可能有严重的bug，参见第6点，总之不建议装
* C++ Intellisense：用的是gtags，本文第一个版本的选择。效果非常非常一般。
* C/C++ Advanced Lint：即cppflylint，本文第二个版本的选择。会产生许多奇怪的警告。总之“过时”了
* Clang-Format：Adapter包含了此功能

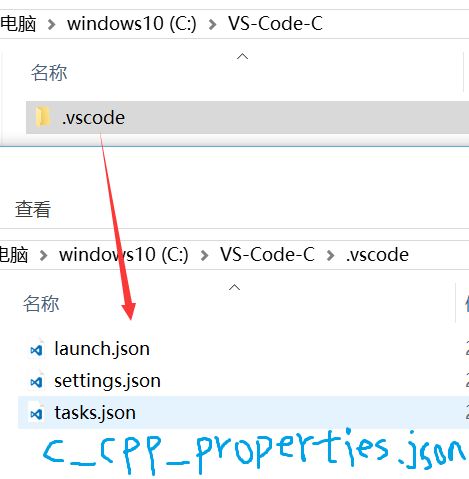
**2. 配置四个.json文件**

此节我当时大部分参考的是@blackkitty的文章，但是现在修改了很多。

先创建一个你打算存放代码的文件夹（称作工作区），**路径不能含有中文和空格和引号**。c语言和c++需要建立不同的工作区（除非你懂得下面json文件的某些选项，则可以做到一个工作区使用不同的build task）。

打开VS Code，**选打开文件夹**（不要选“添加工作区文件夹”，理由见上一句），选择刚才那个文件夹，点VS Code上的新建文件夹，名称为*.vscode*（这样做的原因是Windows的Explorer不允许创建的文件夹第一个字符是点），然后创建 launch.json，tasks.json，settings.json，c\_cpp\_properties.json放到.vscode文件夹下，效果图：

<img src="https://pic4.zhimg.com/v2-f258256ae42b04e30a996f86a3676aa3\_b.jpg" data-caption="" data-size="normal" data-rawwidth="428" data-rawheight="164" class="origin\_image zh-lightbox-thumb" width="428" data-original="https://pic4.zhimg.com/v2-f258256ae42b04e30a996f86a3676aa3\_r.jpg">

<img src="https://pic1.zhimg.com/v2-526f46888adffe5b165f26244d193ce8\_b.jpg" data-caption="" data-size="normal" data-rawwidth="469" data-rawheight="479" class="origin\_image zh-lightbox-thumb" width="469" data-original="https://pic1.zhimg.com/v2-526f46888adffe5b165f26244d193ce8\_r.jpg">

复制以下代码时不要用ie打开本网页。（不过现在知乎已经完全不让ie访问了）复制出来以后，知乎会自动在前面加上几行保留所有权利的字，实际使用的时候肯定要删了的。

*launch.json*代码：

*stopAtEntry*可根据自己喜好修改；cwd可以控制程序运行时的相对路径，如有需要可以改为*${fileDirname}*（感谢

[@xhx](//www.zhihu.com/people/dd30faca74afd3e29376d2226d44475d)

）。其他无需更改，除非你不用windows，则可以用lldb调试（需要自己装）。type和request不变色是正常现象。

// https://github.com/Microsoft/vscode-cpptools/blob/master/launch.md

{

"version": "0.2.0",

"configurations": [

{

"name": "(gdb) Launch", // 配置名称，将会在启动配置的下拉菜单中显示

"type": "cppdbg", // 配置类型，这里只能为cppdbg

"request": "launch", // 请求配置类型，可以为launch（启动）或attach（附加）

"program": "${fileDirname}/${fileBasenameNoExtension}.exe", // 将要进行调试的程序的路径

"args": [], // 程序调试时传递给程序的命令行参数，一般设为空即可

"stopAtEntry": false, // 设为true时程序将暂停在程序入口处，我一般设置为true

"cwd": "${workspaceFolder}", // 调试程序时的工作目录

"environment": [], // （环境变量？）

"externalConsole": true, // 调试时是否显示控制台窗口，一般设置为true显示控制台

"internalConsoleOptions": "neverOpen", // 如果不设为neverOpen，调试时会跳到“调试控制台”选项卡，你应该不需要对gdb手动输命令吧？

"MIMode": "gdb", // 指定连接的调试器，可以为gdb或lldb。但目前lldb在windows下没有预编译好的版本。

"miDebuggerPath": "gdb.exe", // 调试器路径，Windows下后缀不能省略，Linux下则去掉

"setupCommands": [ // 用处未知，模板如此

{

"description": "Enable pretty-printing for gdb",

"text": "-enable-pretty-printing",

"ignoreFailures": false

}

],

"preLaunchTask": "Compile" // 调试会话开始前执行的任务，一般为编译程序。与tasks.json的label相对应

}

]

}

*tasks.json*代码：

*reveal*可根据自己喜好修改，即使设为never，也只是编译时不跳转到“终端”而已，手动点进去还是可以看到，我个人设为never。

命令行参数方面，**-std根据自己的需要修改**。如果使用Clang**编写C语言**，把*command*的值改成clang。如果使用MinGW，编译C用gcc，编译c++用g++，并把-target和-fcolor那两条删去。如果不想要额外警告，把-Wall那一条删去。参数的作用我加了注释，还看不懂，百度gcc使用教程。

// https://code.visualstudio.com/docs/editor/tasks

{

"version": "2.0.0",

"tasks": [

{

"label": "Compile", // 任务名称，与launch.json的preLaunchTask相对应

"command": "clang++", // 要使用的编译器

"args": [

"${file}",

"-o", // 指定输出文件名，不加该参数则默认输出a.exe，Linux下默认a.out

"${fileDirname}/${fileBasenameNoExtension}.exe",

"-g", // 生成和调试有关的信息

"-Wall", // 开启额外警告

"-static-libgcc", // 静态链接

"-fcolor-diagnostics", // 彩色的错误信息？但貌似clang默认开启而gcc不接受此参数

"--target=x86\_64-w64-mingw", // clang的默认target为msvc，不加这一条就会找不到头文件；Linux下去掉这一条

"-std=c++17" // C语言最新标准为c11，或根据自己的需要进行修改

], // 编译命令参数

"type": "shell", // 可以为shell或process，前者相当于先打开shell再输入命令，后者是直接运行命令

"group": {

"kind": "build",

"isDefault": true // 设为false可做到一个tasks.json配置多个编译指令，需要自己修改本文件，我这里不多提

},

"presentation": {

"echo": true,

"reveal": "always", // 在“终端”中显示编译信息的策略，可以为always，silent，never。具体参见VSC的文档

"focus": false, // 设为true后可以使执行task时焦点聚集在终端，但对编译c和c++来说，设为true没有意义

"panel": "shared" // 不同的文件的编译信息共享一个终端面板

}

// "problemMatcher":"$gcc" // 如果你不使用clang，去掉前面的注释符，并在上一条之后加个逗号。照着我的教程做的不需要改（也可以把这行删去)

}

]

}

*settings.json*代码：

把这个文件里的东西放到“用户设置”里也可以覆盖全局设置，自己进行选择。

Code Runner的命令行和某些选项可以根据自己的需要在此处修改，用法还是参见此扩展的文档和百度gcc使用教程。

如果你要使用其他地方的头文件和库文件，可能要往clang.cflags和clang.cxxflags里加-I和-L，用法百度gcc使用教程。

clang的补全，在我过去的测试过程中会让VSC非常卡，但是现在好像没有这个问题了。如果你卡，就把clang的补全关掉，用cpptools的（不需要我指明分别是哪两个吧？）。

Linux下去掉code runner和flags的--target那一条，共四个。

感谢

[@Wellin Boss](//www.zhihu.com/people/e011194994d3415968b3886ade2b588c)

提到的snippetSuggestions。

{

"files.defaultLanguage": "cpp", // ctrl+N新建文件后默认的语言

"editor.formatOnType": true, // 输入时就进行格式化，默认触发字符较少，分号可以触发

"editor.snippetSuggestions": "top", // snippets代码优先显示补全

"code-runner.runInTerminal": true, // 设置成false会在“输出”中输出，无法输入

"code-runner.executorMap": {

"c": "cd $dir && clang $fileName -o $fileNameWithoutExt.exe -Wall -g -Og -static-libgcc -fcolor-diagnostics --target=x86\_64-w64-mingw -std=c11 && $dir$fileNameWithoutExt",

"cpp": "cd $dir && clang++ $fileName -o $fileNameWithoutExt.exe -Wall -g -Og -static-libgcc -fcolor-diagnostics --target=x86\_64-w64-mingw -std=c++17 && $dir$fileNameWithoutExt"

}, // 设置code runner的命令行

"code-runner.saveFileBeforeRun": true, // run code前保存

"code-runner.preserveFocus": true, // 若为false，run code后光标会聚焦到终端上。如果需要频繁输入数据可设为false

"code-runner.clearPreviousOutput": false, // 每次run code前清空属于code runner的终端消息

"C\_Cpp.clang\_format\_sortIncludes": true, // 格式化时调整include的顺序（按字母排序）

"C\_Cpp.intelliSenseEngine": "Default", // 可以为Default或Tag Parser，后者较老，功能较简单。具体差别参考cpptools扩展文档

"C\_Cpp.errorSquiggles": "Disabled", // 因为有clang的lint，所以关掉

"C\_Cpp.autocomplete": "Disabled", // 因为有clang的补全，所以关掉

"clang.cflags": [ // 控制c语言静态检测的参数

"--target=x86\_64-w64-mingw",

"-std=c11",

"-Wall"

],

"clang.cxxflags": [ // 控制c++静态检测时的参数

"--target=x86\_64-w64-mingw",

"-std=c++17",

"-Wall"

],

"clang.completion.enable":true // 效果效果比cpptools要好

}

*c\_cpp\_properties.json*代码：

此文件内容来自于[Microsoft/vscode-cpptools](https://link.zhihu.com/?target=https%3A//github.com/Microsoft/vscode-cpptools/blob/master/Documentation/LanguageServer/MinGW.md)；这个json不允许有注释（其实按照标准本来就不能有）。

如果你没有合并Clang和MinGW，则该文件中的compilerPath**必需**修改成MinGW的**完整**路径，精确到gcc.exe，否则会提示找不到头文件；Linux下应该是/usr/bin/gcc。

如果你自己编写了头文件又不在workspaceFolder下，路径也要加到includePath和browse里。这些路径是否递归有效暂时未知，我的测试是有效的。

Windows下的路径为反斜杠，原本应使用两个反斜杠来转义，但直接用斜杠在VS Code中也接受。

When you set the compilerPath property and change intelliSenseMode to clang-x64, you no longer need to copy the system include path or defines to includePath, browse.path, or defines to enable IntelliSense to work properly.

{

"configurations": [

{

"name": "MinGW",

"intelliSenseMode": "clang-x64",

"compilerPath": "C:/LLVM/bin/gcc.exe",

"includePath": [

"${workspaceFolder}"

],

"defines": [],

"browse": {

"path": [

"${workspaceFolder}"

],

"limitSymbolsToIncludedHeaders": true,

"databaseFilename": ""

},

"cStandard": "c11",

"cppStandard": "c++17"

}

],

"version": 4

}

为什么要往json里写这么多的东西？因为VSC本身并没有对C语言特别优待，对其他许多语言也是这样。另外稍微提一下，以$开头的是VSC预定义的变量，具体参见：[Variables Reference](https://link.zhihu.com/?target=https%3A//code.visualstudio.com/docs/editor/variables-reference" \t "_blank)。

**3. 写代码，编译，调试**

新建文件后就可以写代码了，c语言源代码后缀是.c，c++是.cpp或.C（这也要我教吗……）。代码文件在保存工作区内都可以（一级目录或者自己建立文件夹），**不必**放到.vscode文件夹里，但是还是前面的话，不要含有中文和空格和引号。按Alt+Shift+F（或者右键）可以格式化代码。

停止输入一小段时间（一秒）后就会有Lint，扩展会给一些建议性的warning（比如声明了变量但不使用），自己清楚就行。如果觉得不爽，也有方法不让它提示，比如去掉-Wall就会少一些。如果还想去掉更多的警告，自己找查怎么做，我提示一下：-Wno-...。找好参数后加到clang.cflags、clang.cxxflags和tasks.json的args里。

按ctrl+shift+B单纯编译，按F5为运行并调试（运行前会自动编译）；本来ctrl+F5为运行但不调试，但是在C中貌似没有用，还是会调试。在写程序初期，我强烈建议**不要把f5当作编译来使用**，因为有的bug只会产生警告，不会阻止编译，但这些东西越早期解决越好。编译信息会在底下的“终端”面板里，如果代码有错误，点进去可以看clang报的信息，但因为有Lint了，所以可以轻松很多。

加断点在列号前面点一下就行，如果想从一开始就停下来，可以加在main函数那里，或者*launch.json*中设置*"stopAtEntry": true*。按f11可以一步一步进行，箭头所指的那行代码就是**下一步要运行的代码**。左边有个调试栏，可以看到变量的值,自动栏没有的可以手动添加表达式；把鼠标放到变量上可以看到变量的值，但是只能识别简单的表达式；栈帧对于递归很有用；在某些时候还可以抓取“异常”。

如果你不需要调试，可以直接右键选run code。如果在终端里运行，可以输入数据，但是少了显示时间的功能；在“输出”中则上面两项相反。用它还可以在非工作区内编译运行程序，但executorMap记得放到全局设置里。在终端中按ctrl + C可以终止程序运行。但它其实只是帮你手动输命令，功能并不强，算是适用场景不同吧。另外，楼下的答主韩骏就是此插件作者，有事统统找他（滑稽）。

另外如果按照我的配置，task和code runner还有一点不同的是working directory。前者是你打开的文件夹，后者是文件所在的文件夹。当然它们也都可以自己修改。

如果你想进行少量的多文件编译，对于c语言请使用clang(gcc)把所有文件编译成.o的中间代码，再用clang++(g++)链接.o文件，（为了方便）并把这些命令写进批处理中；这个操作门槛非常低，如果不会，百度gcc使用教程。如果你想进行大量的多文件编译，请学习如何写makefile或使用cmake，并且修改tasks.json的command和args；这个稍微有一点难度。

如果你用VSC还做别的事（比如写前端），或者有不止一个工作区，可以创建一个快捷方式（右键新建），把工作区路径作为参数传给VSC主程序，还可以加个图标。这操作不难，记得打双引号就行。现在1.18有了一个窗口多个工作区的功能，“文件”菜单里也有“保存工作区”这个功能。

<img src="https://pic4.zhimg.com/v2-670bee049e796527501ab8e1dde75a8f\_b.jpg" data-caption="" data-size="normal" data-rawwidth="542" data-rawheight="316" data-default-watermark-src="https://pic4.zhimg.com/v2-670bee049e796527501ab8e1dde75a8f\_b.jpg" class="origin\_image zh-lightbox-thumb" width="542" data-original="https://pic4.zhimg.com/v2-670bee049e796527501ab8e1dde75a8f\_r.jpg">

某些可能出现的错误：

* 如果你只写了个hello world，不加任何断点，**按f5以后黑框框一闪而过是正常现象**。想让程序暂停运行可以在末尾加上一个或两个getchar();，不明白为什么有时要用两个？去问你们C语言老师；或用system("pause")，或加断
* 如果你要进行调试，不要开优化。gcc用-Og还可以保留一些调试信息，但clang用了以后就不能用gdb调试了。即使如此我还是在某一次写代码的时候遇到了无法跳入函数的问题，而VS可以跳入
* 重命名文件后，原来已有的Lint还会在问题栏里；修改了文件后断点可能会失效。以及还存在一些其他的像这样的小bug，一般关掉VSC再开就行
* preLaunchTask“Compile”已终止，退出代码为 1：编译有error并且你用的是F5运行的就会有这个提示，有warning是否会触发不清楚；如果没有error，点仍然调试就行了；如果有error你还点仍然调试，**就会调试上一次编译成功的文件**。有一种原因是原程序仍在运行，无法被覆盖（比如死循环），终端里报错为permission denied，任务管理器结束那个进程即可。但其实所有的编译失败都会触发这个错误，出错的返回值是1难道不是常识？所以仅仅告诉我出现了这个提示**根本没用**，因为它的意思就是出错了，没有人能看出原因。这也是为什么我要强烈建议不要把F5当作编译来使用，按F5出了问题，我根本看不出是编译期有问题还是调试期有问题，或是你自己的代码有问题
* 无法打开...，找不到文件(file:///build/glibc-OTsEL5/glibc-2.27/...)：我在Linux下遇到了这个问题，下一个glibc放到指定位置就行，wget http://ftp.gnu.org/gnu/glibc/glibc-2.27.tar.xz，剩下的就不要问我了。或者参见这个：[Disable "Unable to open file" during debug · Issue #811 · Microsoft/vscode-cpptools](https://link.zhihu.com/?target=https%3A//github.com/Microsoft/vscode-cpptools/issues/811%23issuecomment-376658727)

**4. 其他设置**

我的一些其他的设置，用在全局settings.json里，根据自己的情况调整，不需要全部照着我的写。**写完一个以后要打逗号，最后一个就不用了**。

"editor.fontFamily": "等距更纱黑体 SC", // 控制编辑器字体

"workbench.colorTheme": "One Dark Pro", // 主题

"files.trimTrailingWhitespace": true, // 保存时，删除每一行末尾的空格

"workbench.colorCustomizations": {

"activityBar.foreground": "#39C5BB" // 自定义颜色

},

"git.enabled": false, // 如果你不用git，我建议你关闭它

"git.ignoreMissingGitWarning": true, // 同上

"editor.minimap.enabled": false, // 我个人不用minimap，就是右边那个东西

"editor.dragAndDrop": false, // 选中文字后，可以拖动它们调整位置。我是不需要

"files.autoGuessEncoding": false, // 启用后，会在打开文件时尝试猜测字符集编码。我关闭的理由见6

"[c]": {

// "files.encoding": "gbk" // 这样的格式可以对指定后缀的文件应用设置，如果你实在想用gbk，就这样设置吧。cpp同理。

}

更纱黑体是楼下B神做的字体，特点是标点好看（误）：[be5invis/Sarasa-Gothic](https://link.zhihu.com/?target=https%3A//github.com/be5invis/Sarasa-Gothic" \t "_blank)

Consolas虽然是Windows自带字体中还算行的，但它只有英文字体；微软雅黑虽然是非衬线字体，但它不是等距的，这一点非常不适合编程，等线也不等距；中易宋体……告辞。不下新的字体，其他两大系统我不清楚，Windows下简直没有编程可用的字体。

**5. 进一步学习**

* 学好英语，然后可以阅读官方英文文档：[Documentation for Visual Studio Code](https://link.zhihu.com/?target=https%3A//code.visualstudio.com/docs" \t "_blank)
* 快捷键：[vscode: Visual Studio Code 常用快捷键](https://link.zhihu.com/?target=http%3A//www.cnblogs.com/bindong/p/6045957.html" \t "_blank)  
  英文文档中当然有快捷键的信息，而且英文文档会更新。这个单独列出来还是给初学者吧。  
  我就提示一点特别重要的：出现Intellisense或者snippets的时候按tab可以补全代码。
* VS Code其实是前端利器，学html, css, javascript时可以好好利用哦。

**6. 关于乱码**

VS Code输出会出现乱码，很多人都遇到过。这是因为VS Code内部用的是utf-8编码，cmd/Powershell是gbk编码。直接编译，会把“你好”输出成“浣犲ソ”。如果把cmd的活动代码页改成65001，会出现汉字只能显示一半的问题，而且怎么修改成UTF8还是个问题（这个可以参见楼下

[@Zombie110year](//www.zhihu.com/people/a1cd971efb1c7c3ca3d28871ce7509b8)

同学的测试）。Linux就没有这个问题。如果你只是想在VSC里运行而且不在别人的WIndows上运行，可以考虑使用Powershell Core。

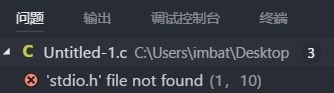
原本的解决方法是使用gcc，编译时用*-fexec-charset=GBK*这个参数，生成的程序就是GBK编码的*。*但是，clang的execution-charset supports only UTF-8。所以，生成的程序在cmd/ps以及VS Code的终端（其实也是powershell）中运行，输出中文还是会乱码；但是在VS Code的“输出”中就是正常的。如果想解决这个问题，可以百度“宽字符输出”，或者自己手动在cmd里用gcc加上上面那个参数编译一遍（可以写个批处理）。

如果是打开已有的以GBK编码的文件，VS Code默认会以UTF-8编码打开（除非你设置了猜测编码），这样编辑器内的中文就会乱码，不过对于初学C的同学来说，写的代码一般只有注释是中文。此时要点右下角的GBK，选“通过编码重新打开”，选UTF-8即可。GBKtoUTF8这个扩展，理论上如果VSC检测出的是GBK编码的，它就会自动做“以UTF-8格式保存”这个操作；但是如果VSC没有检测出是GBK编码，它就什么也不会做。但是貌似它有bug，会把当前文件复制一遍插入到光标处……所以不推荐使用

如果你没有注意到一个GBK编码的文件被VSC以UTF-8的编码打开了，又进行了保存，按照我的测试，这文件里的中文应该是找不回来了。这个还是比较危险的。而且如果打开了编码猜测，VSC又猜错了的话……所以我是关闭编码自动猜测的。中文特别少的时候猜错几率很大。

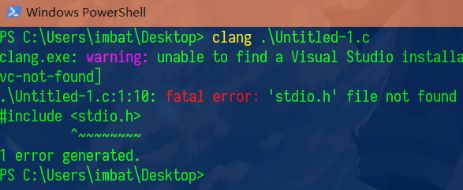
这样做了以后，在含有中文的路径下可以编译，但是仍然不能调试，所以还是把代码放到**不含中文的路径**中吧。如果把代码文件发给其他用Windows的人，最好转成gbk，否则别人用记事本打开有可能会乱码（不过貌似1709改进了记事本的编码猜测，1803的下一个版本连LF都支持了）。

**7. 找不到头文件的错误**

<img src="https://pic3.zhimg.com/v2-877058be4208e6b3648a1da6cec6205e\_b.jpg" data-caption="" data-size="normal" data-rawwidth="334" data-rawheight="93" class="content\_image" width="334">

有几位同学遇到了路径设置正确，编译也通过，但是“问题"面板里出现找不到头文件的error。我也遇到过。这个error是cpptools报的。可能的解决方法是把你需要的头文件的路径加到c\_cpp\_properties.json中，或者你的compilerPath没有设置正确。如果还是解决不了，反正不影响编译，就当做没看到算了。如果你遇到了又解决了可以留言告诉大家。如果是非工作区选c语言或者c++，出现这个错误很正常，因为不满足前提：路径设置正确（没有c\_cpp\_properties.json）。

**还有一种可能，看评论区BladLust同学的回复。**

<img src="https://pic2.zhimg.com/v2-a7e1578ac1d50bcba9ccc3986ae607ad\_b.jpg" data-caption="" data-size="normal" data-rawwidth="463" data-rawheight="190" class="origin\_image zh-lightbox-thumb" width="463" data-original="https://pic2.zhimg.com/v2-a7e1578ac1d50bcba9ccc3986ae607ad\_r.jpg">

如果是这个错误，这是因为clang的默认target为msvc，需要加--target=x86\_64-w64-mingw这个参数才行。这个默认target貌似是写死在源代码里的，反正我找了一圈是没找到正常修改办法，下载clang的源代码，自己改掉，再编译clang本身，也许可以解决。或者装Windows sdk而不使用mingw，这样就符合默认target了，参考第九点。当然最简单的办法就是用gcc。

**8. 其他**

* 按照这样配置，长期编译代码下来肯定有一大堆的exe，还可能分散在不同的文件夹里。你可以考虑修改一下json文件，把生成文件的目录指定到一个专门放exe的文件夹里（如果不会，百度gcc使用教程）。或者资源管理器右上角搜索\*.exe，就可以搜出它们。或者写个bat删了。都很简单。
* json是一种数据交换格式，大部分是JavaScript的子集，数据冗余度小。VSC和各个扩展会读取json中的条目，来决定某些功能的行为。这么多条目哪里来的呢？这其实和API差不多。扩展开发者会把允许修改的选项“告诉”VSC，各个扩展的安装页面都有写，VSC又有intellisense，所以其实很容易写。如果是单纯使用json，我觉得就算从来没有见过，边看边猜也能写个大概。又因为扩展开源，你甚至可以去扩展的github页面和开发者聊天。
* Windows 10，默认输入法只有一个微软拼音，按一次shift就能进行中英转换，而为了保持兼容，按ctrl加空格也能进行中英转换，而这个快捷键正是强制触发Intellisense的快捷键。所以，我强烈建议手动添加“英语”语言输入法，写非前端代码时切换到纯英文输入法（win+空格）。这样也可以解决某些游戏需要用到shift键但是同样快捷键冲突的问题。具体操作我就不说了，自己百度。
* VSC是集成git的，不过对于初学者可能并不会用到。我在用某一个版本时，看到git提示我有文件发生了改变。我想消掉这个提示，乱点点了discard changes，然后它就把我的工作区清空了……后来我就把它关了。至于怎么用git，那又是另一个话题了，慢慢学吧。
* tasks.json中的"problemMatcher":"$gcc"会解析终端中的错误提示，因为已经有Clang的Lint了，就不需要这个；如果用了Clang Command Adapter又打开这个，则会出现双重错误提示。本来1.11就说可以写$gcc的，但当时其实并不支持，现在早就能用了。不过如果要用非预设版本，就需要自己写了。
* 想在自带的终端里进行调试？Follow：[Use the VSCode debugConsole instead of externalConsole · Issue #35 · Microsoft/vscode-cpptools](https://link.zhihu.com/?target=https%3A//github.com/Microsoft/vscode-cpptools/issues/35" \t "_blank)

**9. 其他工具链的选择**

* 使用MinGW编译但仍用Clang提供Lint：tasks.json的命令行自己改一改，code runner的命令行在settings.json里，自己改。这样可以在终端中输出不乱码，参考第六点。缺点：编译用的不是Clang，编译速度相对慢。Lint可能提示的警告不全，比如Clang给出的"did you mean ..."提示，Lint就可能捕获不到
* MinGW-w64 + 官方扩展：不使用Clang。除了上面做的，tasks.json里problemMatcher打开；settings.json里的东西自己改一改。缺点：Windows下的Lint效果真的真的很差，Linux稍微好一点。感觉相比上一个方案没有优点？
* Windows SDK + 官方扩展：VS Installer选VC++工具集和一个完整的SDK（默认勾上的那个就是）即可。扩展用cpptools，c\_cpp\_properties.json可以自动化配置（ctrl+shift+p, edit configurations）；另外两个json也要改，VS的编译器是cl，参数也要改；调试器也许可以用VS的。不过这样我觉得也许还不如直接用VS，而且我没试过
* Clang + Windows SDK + C/C++ Clang Command Adapter：这套方案需要修改的东西不多，因为编译用的仍然是Clang。c\_cpp\_properties.json还是可以自动化配置的；各个地方删去--target那个参数，因为头文件用的不是MinGW提供的，默认用的就是MSVC的。VS Installer里还有一个Clang/c2，根据龚大的文章这个有坑，所以装官方的版本吧。还是同上一条，感觉不如直接用VS
* 如果以上都看不懂，可以试试这个配置好的（不过人家的配置方法和我的不一样）：[【VSCode】Windows下VSCode便携式c/c++环境](https://link.zhihu.com/?target=https%3A//blog.csdn.net/c_duoduo/article/details/52083494" \t "_blank)
* 如果不想用VSC写了，可以看看这篇问题：[毫无编程基础的小白准备学习C语言，用VC6还是VS2015？](https://www.zhihu.com/question/40929777)
* codeblocks现在还活着，论坛里有nightly build，配置一番（虽然同样有点折腾）也可用。Clion界面美观，功能应该也挺强，不过只有英文，刚上手用起来可能有点困难，学生可以免费申请key，否则收费

**10. 我编写代码的体验**

体积上，合并后的llvm文件夹占1.3g，vsc 0.2g，加上一些扩展。如果只是用来写c，可能体积占用并不算小。内存占用，如果VSC不出bug，还是比较少的（0.5g左右）。

VSC的第一优势也许是好看？其实用它来写C优势没有想象中的那么大，不过至少比wintc、cfree、dev c++强。Lint真的太重要了。

不过我有一点想对学生党说：能自己百度到这篇文章，努力去看懂、动手配置，已经比贴吧无数伸手党和等着老师在qq群里发ide的人强了很多了。另外如果有能力，我还是建议你们读读VSC的文档，并不复杂，体验一下英语的实际应用也不错哦。